

Virtual Representations for an Iterative IoT Deployment

Sebastian R. Bader

Karlsruhe Institute of Technology
Karlsruhe, Germany

Maria Maleshkova

Karlsruhe Institute of Technology
Karlsruhe, Germany

ABSTRACT

A central vision of the Internet of Things is the representation of the physical world in a consistent virtual environment. Especially in the context of smart factories the connection of the different, heterogeneous production modules through a digital shop floor promises faster conversion rates, data-driven maintenance or automated machine configurations for use cases which haven't been recognized at design time. Nevertheless, these scenarios demand IoT representations of all participating machines and components, which requires high installation efforts and hardware adjustments.

We propose an incremental process for bringing the shop floor closer to the IoT vision. Currently the majority of systems, components or parts are not yet connected with the internet and might not even provide the possibility to be technically equipped with sensors. However, those could be essential parts for a realistic digital shop floor representation. We therefore propose *Virtual Representations*, which are capable of independently calculating a physical object's condition by dynamically collecting and interpreting already available data through RESTful Web APIs. The internal logic of such *Virtual Representations* are further adjustable at runtime since changes to its respective physical object, its environment or updates to the resource itself should not cause any downtime.

KEYWORDS

internet of things; linked data; virtual integration

ACM Reference Format:

Sebastian R. Bader and Maria Maleshkova. 2018. Virtual Representations for an Iterative IoT Deployment. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3184558.3191657>

1 INTRODUCTION

The ongoing digitization and interconnecting of devices of any kind through internet technologies are a prominent current trend [15]. The Internet of Things (IoT) comprises these efforts to represent physical and virtual objects through common patterns in order to allow a digital information exchange between them. Gubbi et al., similarly to others, define the IoT as an "Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing common operating picture for enabling innovative applications. This is achieved by seamless large scale sensing, data analytics and information representation using cutting edge ubiquitous sensing

and cloud computing." [7]. Obviously, the regarded 'things' are restricted to physical objects, equipped with sensors or actuators, and with some kind of connection capability (referred to as 'devices').

We claim that the installation of sensors, the monitoring of components or the connection in digital networks is not a goal in itself but is always demand-driven. Consequently, a certain need is defined in the first place (e.g. digitally monitoring a machine's abrasion state), creating requirements on the data side. If the required data is not yet measured, additional sensors (or actuators) at the physical objects are taken into account. Only if the added value justifies this investment the hardware is updated. Following this argumentation, the creation of an IoT setting especially in the industrial domain is an iterative process, where more and more things get a digital counterpart but a complete coverage of all things does not happen. In contrast to that, the key potential of the Internet of Things is not only the integration of such devices and then conducting a defined analysis but enabling previously *unknown* applications.

Especially in the context of smart factories notable efforts are conducted to bring the diverse and heterogeneous components of the shop floor to a consistent digital layer. In order to keep the integration efforts sustainable an easy to understand data and interaction model are required. The self-explaining characteristics and the high maturity level of the Semantic Web make it a suitable candidate for a future-proven solution. Such an integration layer (e.g. the so called 'Implementation Viewpoint' in Fig. 1) encapsulates the complexity of underlying network implementation. Generally aspired goals are e.g. faster conversions, data-driven maintenance or automated machine configurations in order to reduce down times and therefore increase the overall efficiency achieved by the overlying applications.

Even though IoT technologies support the integration process, the IoT is not only about increasing the flexibility or new insights into the behaviour of one machine but of the production line or even supply chain as a whole. Applications achieving these aims need to be deployed rapidly if not automatically with minimal effort. A heterogeneous landscape with a use case-driven integration approach requires repetitive efforts, contradicting any potentially acquired efficiency gains. The main reason is that the configuration of interfaces is usually oriented towards scenarios relevant at the respective point in time but not for generic requirements of such later deployed systems. As the input needs of future applications are per se unknown at design time, a consistent and comprehensive digital reflection of the real world is more efficient and sustainable. Achieving this flexibility is the main benefit where the IoT can contribute to the manufacturing industry.

Consequently, nearly full information about the current state of all things must be reflected: the relevant machines, their respective components as well as the involved materials and products. Whereas many critical machine parts are not observable with economically reasonable investment (accessibility, unreliable network

This paper is published under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. Authors reserve their rights to disseminate the work on their personal and corporate Web sites with the appropriate attribution.

WWW '18 Companion, April 23–27, 2018, Lyon, France

© 2018 IW3C2 (International World Wide Web Conference Committee), published under Creative Commons CC BY 4.0 License.

ACM ISBN 978-1-4503-5640-4/18/04.

<https://doi.org/10.1145/3184558.3191657>

access, high update costs) a continuous and complete monitoring of any system, sub-system and sub-sub-system from one day to the other is unrealistic anyway.

We propose an incremental process for bringing the shop floor to the IoT. Many machines and devices nowadays are already (partly) IoT-ready. However, systems, modules or parts, which cannot be connected to the internet or promptly equipped with sensors are also required for a true digital shop floor. We, therefore, develop **Virtual Representations**, a digital representation which does not necessarily require a direct network connection to the physical thing. The *Virtual Representation* is responsible for independently calculating the object's condition by collecting and interpreting already available environment data. We present a RESTful interaction model to deploy and scale *Virtual Representations*. Furthermore, we show how the configuration specifying a *Virtual Representation* can be adapted at runtime by RESTful Web APIs, since changes in the real world need to be reflected in a transparent and standardized manner. We implemented the approach based on a Linked Data Platform server for future extension. A demonstrating show case is also publicly available to illustrate the concept. We argue that the significantly lower cost of a *Virtual Representation* based on omitted hardware updates (if those are possible at all) motivates the use of *Virtual Representation's* and of simulated values.

One example application scenario might be a robot gripper arm, as one unit in an industrial production line. Its motors and the supplied material are directly observable by appropriate analytic components. In contrast, e.g. the shafts transmitting the force from the motor to the jaws and the jaws themselves cannot be monitored as its movements and the robot's configuration do not allow a sufficient measurement. Nevertheless, the abrasion state of both components might be a critical factor for a maintenance strategy. A *Virtual Representation* can encapsulate the necessary data collection (number and duration of loads, applied material type etc.) and the currently best performing evaluation logic. The thereby encapsulated *Virtual Representation* can be shifted easily, connected to other cyber-physical systems or updated as it confirms to well established Web standards. This allows a sufficiently detailed and flexibly modelled smart factory with a minimum required investment.

The paper is structured as follows: Section 2 gives a brief overview on related concepts discussed in the Web and IoT community. Section 3 further illustrates the use case followed by the conceptional design of *Virtual Representations* in Section 4. Section 5 summarizes the proposed methods and outlines future implications.

2 RELATED WORK

A Thing in the most general view can be either a physical object, a software program, a piece of information or any other kind of identifiable resource. Within the IoT, more and more things become part of the internet – at the moment mainly focusing on production related objects (e.g. machines, materials etc.) or Smart Homes (smart/connected furniture). To establish a digital information exchange, a virtual resource is created in order to identify and access those objects. These resources need to be identifiable by a unique key and accessible through a commonly understood locator. Both purposes are served by URIs and their widespread usage throughout the internet makes them the identifier of choice.

The internet and in particular the World Wide Web offers already a well understood and widely accepted infrastructure to exchange data. Well-established Web technologies like URIs, HTTP and hyperlinks have proven to allow easy and reliable communication mechanisms in a decentralized manner. Cloud services and on demand Web solutions offer fast and flexible deployment of applications, a strict requirement for a smart factory. Maleshkova et al. analyzed the state of deployed Web APIs and RESTful services [13]. They conclude that a significant ratio of APIs lacks sufficient descriptions and at the same time miss necessary information on both the input and output data sets. The Semantic Web on the other hand adds meaning to data objects and can reduce the mentioned integration effort. However, thereby created Web of Things does neither specify the interaction patterns of the regarded things nor does it model the intended relationship with the physical world.

Perera et al. write about the Avatar concept as an interoperability concept between objects 'using standard protocols and technologies defined by the W3C for the Web, coupled with a distributed service-oriented mediation infrastructure' [14]. Thus, avatars can serve as a representation for both physical and software things but do not contain formalisms to simulate unconnected objects.

'Physical entities' and 'virtual entities' are basic concepts in the domain of cyber-physical systems. Lee et al. introduce a general five layered architecture for cyber-physical systems. Their cyber layer includes 'cyber twins' which capture and preprocess the captured data for higher level applications [11]. The thereby created virtual modeling of the factory and its entities are the consistent data suppliers for the factory management. Unfortunately, they do not consider the enormous necessary effort to fully represent all machines of a production line and require to have all entities at every step and do not discuss how an environment with only partly digitized machines can benefit from their vision.

Glaessgen and Stargel formulate requirements on so called digital twins regarding NASA and US air force vehicles. Their focus is to simulate any perceived incident to the physical vehicle at the virtual one in order to get a higher accuracy predicting the current state of the vehicle. Tao et al. focus on the product lifecycle (design, manufacturing, service). They identify a research gap in the field of Product Lifecycle Management in the form of a disconnection between the physical object and the virtual information available during the several lifecycle stages. Their digital twin concept follows the definition of [6] and focuses on information presentation in the form of a virtual objects but do not consider any manipulations of those. Therefore, their digital twin concept mainly serves as a virtual model and information container and rather the thing itself as any (virtual) interaction pattern is missing.

The Industrial Internet Consortium aims to introduce common standards for the Industrial Internet of Things. As part of their reference architecture (Fig. 1) the consortium discusses the trend to "base their control decisions on the simulation model rather than a control engineer's equation" [12]. Especially the introduced Functional Viewpoint contains the modeling of things but still only targets objects directly equipped with sensors or actuators. In addition, context information is briefly discussed but limited to the semantically described relationships of the previously mentioned entities.

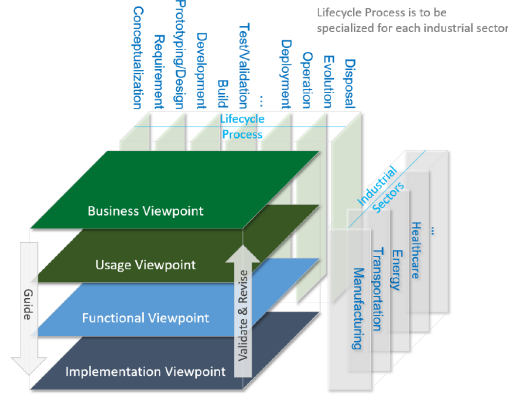


Figure 1: Architectural Framework according to [12]

Similar to the Industrial Internet Consortium the Plattform Industrie 4.0 proposes the 'Asset Administration Shell' containing an identifier, a referent, the referent definition and a set of characteristics [3]. It contains basic descriptions (header) of its provided data objects and functions in the body. An 'Asset Administration Shell' can be seen as a specification of the virtual entity concept from cyber-physical systems but does also only regard connected objects. Even though an 'Asset Administration Shell' can also represent software or a Web service the authors do not discuss any strategies how their general concept also allows the introduction of unconnected things to the digital shop floor.

Edge computing [9], edge analytics or fog computing [4] proposes to conduct substantial part of computing tasks close to the physical thing at the edge of the network. The main advantages are less network traffic and response times as data processing takes place close to the data sources. Even though simulating resources close to their physical location can be seen as some kind of edge computing the paradigm itself lacks – to the best of our knowledge – standardized methods describing how to implement and how to interact with the resources of interest.

The Resource Description Framework (RDF) models resources in the form of triples (subject, predicate, object). Each part of the triple can either be a resource in the form of a globally unique URI or, in case of the object position, also a sequence of characters referred to as Literals. RDF also allows several serialization formats and therefore serves as a flexible data format for both descriptions and productive data. Linked Data further introduces dereferenceable identifiers (HTTP URIs) and relationships to other web resources. Linked Data Platform [16] specifies data manipulation (CRUD) operations on Linked Data resources other the web.

In particular, in the context of the Web, Smart Components [10] paves the way to flexible and context-dependent integration. A Smart Component not only exchanges semantically defined data (through its regular Web API) but also can adjust their program code through a so called meta API, using exactly the same interaction mechanisms enabled by the Linked Data-Fu engine [8]. Self-governed Components [2] utilize this functionality to independently trigger actions and achieve more robust Web service networks.

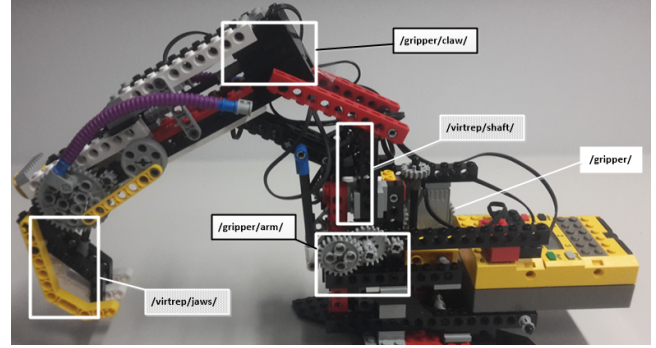


Figure 2: Illustration of IoT resources (white) and Virtual Representations (grey)

3 SCENARIO: ROBOT GRIPPER ARM

An implementation of the previously described scenario is publicly accessible. A Linked Data Platform server hosts the IoT representation of a demonstration gripper as an LDP RDF Source together with its components¹ (Fig. 2). Requesting the web resource of the gripper is possible via GET requests as well as towards its sub-resources, returning the current state of the physical object as far as observed together with some meta data. RESTful interactions with the gripper's Web API are enabled by overwriting the current resource state of the gripper's arm² and claw³ by sending an RDF statement with the predicate *saref:hasState* and the new state ('up'/'down' and 'opened'/'closed') at the object position. Example requests⁴ illustrate the use case.

Even though the gripper itself and two components are digitally accessible it is easy to understand that this only represents a fraction of the actually installed components. In order to e.g. decide of the optimal maintenance strategy further information on other parts is essential too. We illustrate our strategy to bring those parts to the same integration layer by representing them as *Virtual Representations*. Two *Virtual Representations* are hosted on an Apache Marmotta⁵ server representing the shaft⁶ and the jaws⁷.

The *Virtual Representation* simulates the physical object's state. To do so, the relation between the already available input data and the required features needs to be determined. Such an algorithm or heuristic is of course based on the currently most accurate available knowledge. Furthermore, in the same way the setup in the physical world changes or new information on the performance of the regarded components become known, the deployed logic of the *Virtual Representation* needs transparent update functionalities.

Let us assume that the relation between the shaft's abrasion state and the conducted number of actions is a linear function, where the component is still perfectly fine after 10 usages but broken after 20. A straight forward heuristic would be:

¹<http://km.aifb.kit.edu/services/step-iot/gripper/>

²<http://km.aifb.kit.edu/services/step-iot/gripper/arm/>

³<http://km.aifb.kit.edu/services/step-iot/gripper/claw/>

⁴<https://github.com/aifb/virtrep/requests/>

⁵<http://marmotta.apache.org/>

⁶<http://km.aifb.kit.edu/services/bader4/marmotta/ldp/ShaftContainer/shaft/>

⁷<http://km.aifb.kit.edu/services/bader4/marmotta/ldp/JawsContainer/jaws/>

$$\text{abrasion}(\text{actions}) = \begin{cases} 0 & , |\text{actions}| < 10 \\ \min\{\frac{1}{10} * (|\text{actions}| - 10), 1\} & , \text{else} \end{cases} \quad (1)$$

This naive function is most probably not very accurate. After better insights in the abrasion process have revealed a more fine grained relationship like Eq. 2 the formerly deployed heuristic must be adjusted. We show in the following how a *Virtual Representation* can be initialized with Eq. 1 and then be easily updated with Eq. 2.

$$\text{abrasion}(\text{actions}) = \begin{cases} 0 & , |\text{actions}| < 10 \\ \min\{\frac{1}{1000} * (|\text{actions}| - 10)^3, 1\} & , \text{else} \end{cases} \quad (2)$$

4 THE VIRTUAL REPRESENTATION

The *Virtual Representation* at its core contains a) some descriptions of its physical counterpart, b) its current (simulated) state, c) the function/algorithm responsible for the calculation of derived features and a RESTful pattern to request it. A *Virtual Representation* is a Web resource identified by a globally unique URI. In this paper we focus on the interaction pattern and how the derivation function can be implemented, its characteristics and how to interact with it.

We model a *Virtual Representation* as a RESTful Web resource producing RDF, in particular Linked Data⁸. RDF as the common data format has the advantage of being a very mature and wide-spread standard as it serves as a corner stone of the Semantic Web Stack. Its native connection to formal knowledge representations make RDF the data format of choice for a loosely coupled environment like the regarded shop floor with many heterogeneous machines from different vendors.

4.1 Configuration Function for Derived Features

A configuration function f of a *Virtual Representation* is a relation of derived features y (e.g. the current abrasion state) by evaluating currently available input data x (e.g. the number of movements). For this paper, we rely on Linked Data as our data format in order to cope the syntactical and semantic interoperability at the same time. Therefore, both input data and derived features are Linked Data triples $(x_i, y_j \in L)$ with L being the set of RDF triples compliant to the Linked Data principles.

As the Linked Data-Fu engine [8] is used to collect and process the input data x in the demonstration implementation of *Virtual Representations*, the function f needs to be a two stage process. First, a set of declarative rules in N3 syntax defines the initial data and processing steps. We denote this set according to [8] as a program p . Executing a rule of p leads to either an execution of HTTP requests towards RESTful Web APIs – responding with RDF – or the derivation of new RDF statements locally. The second step involves a SPARQL Construct query q filtering the collected data and finally generating y . The set of RDF triples y representing a *Virtual Representation* is therefore for the demonstration implementation:

$$y = f(x) = q \circ p(x) \quad (3)$$

An example program is shown in Fig. 3. Lines 8 to 11 specify two HTTP request available IoT entities as Linked Data resources. The

⁸source code available at <https://github.com/aifb/virtrep/>

```

1 @prefix http: <http://www.w3.org/2011/http#> .
2 @prefix httpm: <http://www.w3.org/2011/http-methods#> .
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
5 @prefix math: <http://www.w3.org/2000/10/swap/math#> .
6
7 # requesting necessary information
8 { [] http:mthd httpm:GET ;
9   http:requestURI <http://km.aifb.kit.edu/services/step-iot/
  gripper/arm/> .}
10 { [] http:mthd httpm:GET ;
11   http:requestURI <http://km.aifb.kit.edu/services/bader4/
  marmotta/ldp/material/> .}
12
13 # calculate abrasion state
14 {
15   ?arm <http://step.aifb.kit.edu/numberOfRequests> ?r .
16   ?r math:notGreaterThan "10" .
17   ?r math:notGreaterThan "20" .
18   (?r "10") math:difference ?diff .
19   (?diff "0.1") math:product ?abrasion .
20
21 } => {
22   ?arm <http://step.aifb.kit.edu/hasAbrasion> ?abrasion .
23 }

```

Figure 3: Integrating data from external sources and deriving new information (see Eq. 1)

rule from line 14 to 23 derives the new statement specifying the current abrasion state as the condition in the body part (lines 15 to 19) of the rule is satisfied by the above requested RDF set. The query in Fig. 4 receives all statements from the program execution and constructs the description of the *Virtual Representation* as specified from line 2 to 10 in the form of a SPARQL Construct query.

One has to note that the chosen form of the configuration function is not a general requirement for *Virtual Representations* but due to the chosen engine. Different implementations can require other formats like e.g. machine learning models or scripts. The proposed configuration method has the main advantage of being familiar with the Semantic Web and therefore form one single technology stack with RDF and Linked Data.

4.2 Interaction Model

A *Virtual Representation* is the combination of meta data triples together with the set of derived triples and a resource representing the function code. As this resource is also a Linked Data Platform Resource (LDP-R), it can be manipulated according to the Linked Data Platform specification. Even though the concept of *Virtual Representations* is conforming to the Linked Data Platform specification, *Virtual Representations* enhance their regular interaction

```

1 CONSTRUCT {
2   </shaft> rdf:type :VirtualRepresentation .
3   </shaft> rdfs:label "shaft" .
4   </shaft> :hasProgram </shaft/program/> .
5   </shaft> :hasQuery </shaft/query/> .
6   </shaft> saref:hasState </shaft/abrasion/> .
7   </shaft> saref:accomplishes ?job .
8
9   <shaft/abrasion> rdf:value ?abrasion .
10  <shaft/abrasion> rdf:type saref:State .
11 } WHERE {
12   ?s saref:hasState ?abrasion .
13   ?t saref:accomplishes ?job .
14 }

```

Figure 4: Creating the *Virtual Representation* through a SPARQL Construct query

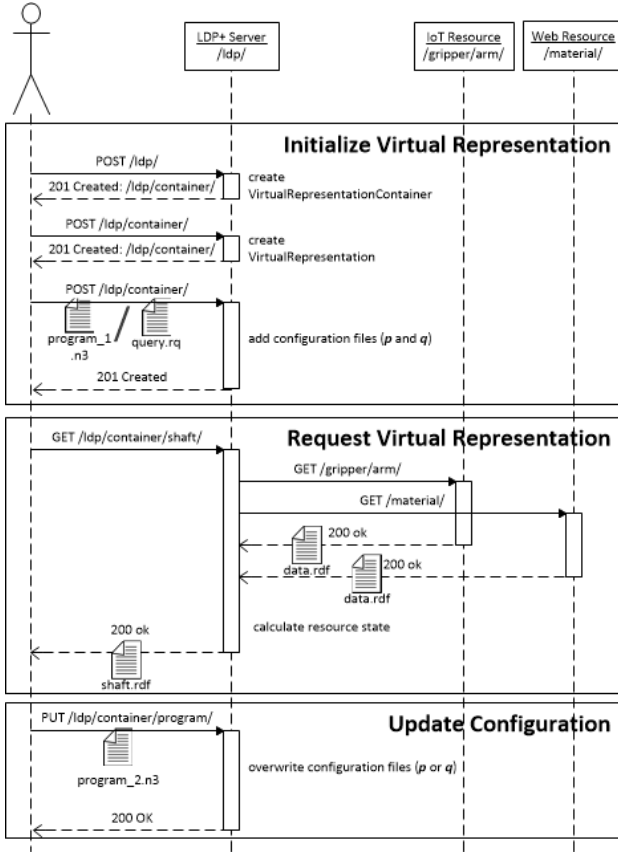


Figure 5: Interaction model for *Virtual Representations*

model by requiring a further configuration. An adaption of a Linked Data Platform server enables the proposed deployment of *Virtual Representations* as shown in Fig. 5.

The Linked Data Platform specification defines how clients can interact with LDP Web Resources in a RESTful manner. We follow the LDP specification regarding the creation, updating and deletion of Web resources. In particular, *Virtual Representations* and the additionally necessary configurations (p and q) are themselves Web resources. More precisely the configurations resources are LDP Non-RDF Sources whereas the *Virtual Representation* itself is an LDP RDF Source. The distinction is necessary as neither the SPARQL syntax of q nor Notation3 of p or any other model are necessarily RDF serializations, a requirement for an LDP RDF Source.

In general, two possible strategies can be applied to create a *Virtual Representation* in a RESTful manner. On the one hand, a user agent can periodically calculate the data, send it to the server which then enables access to the latest received state of the resource. Alternatively, the server waits until a client requests the resource and then calculates the data on the fly. The first approach has advantages in an environment with unreliable network access as at least some version of the requested data is available. Also, storing each state creates a time series and allows historic analysis. Nevertheless, especially in the context of the IoT update rates often in the range of milliseconds are creating enormous data volumes.

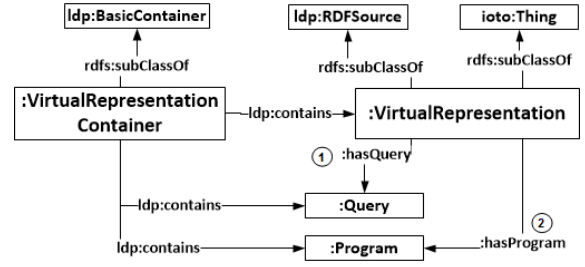


Figure 6: Basic classes and relations to create a *Virtual Representation*

To receive a lightweight representation and also to increase the scalability of the concept we therefore promote the calculation of a *Virtual Representation*'s state on the fly. An HTTP GET request on the *Virtual Representation* triggers the server to load both the program and the query into the engine and executes it (Fig. 5). The dynamically created RDF statements are calculated, representing the related physical object in the same way as a regular Web resource. As we rely on the Linked Data-Fu engine the loading, compilation and execution process happens at real-time and results in a usually not notable overhead.

In order to identify the desired configuration of a *Virtual Representation*, the engine needs additional information. The relations (1) and (2) of the data model (Fig. 6) include these relations. A naive client would expect these statements to reside at the *Virtual Representation* itself as both include the *Virtual Representation* as their subject. Unfortunately, the *Virtual Representation* is calculated dynamically at request time and not stored in the server's database, and therefore not accessible for the engine itself. Even though one could deposit the statements (1) and (2) at any other LDP resource it is better practice to keep such information as close as possible to the related *Virtual Representation*. We therefore introduce the *Virtual Representation Container*, a LDP Basic Container with the additional requirement of containing exactly one *Virtual Representation* together with its configuration, here the query and program resources. The engine can locate the Web resources p and q as the container's child resources and therefore combine the necessary Web resources.

4.3 Categorization of *Virtual Representations*

In REST terms as introduced by [5] the proposed concept of *Virtual Representations* cannot be seen as a user agent. Even though the *Virtual Representation* can initiate request through a client connector it is not the source of the initial request. In the same way the *origin server* concept does not apply as source of the requested thing is not directly connected with it. It is a general problem with the IoT that the virtual reference is not the target of the request which in fact is the physical object. Therefore, any IoT entity must be some kind of intermediary. *Virtual Representations* have server and client components as required for a *proxy* but without allowing a client to choose whether to use it. In contrast, a *gateway* forwards the request but acts as an origin server for the client. *Virtual Representations* act in that way, making them a *gateway* for the digital shop floor without necessarily informing the client about this role.

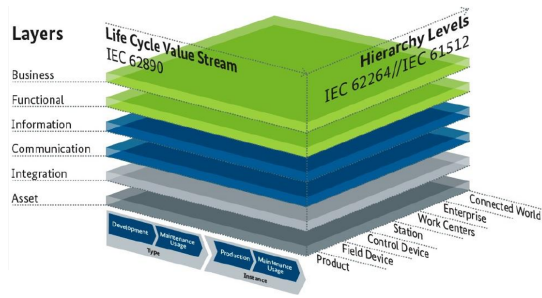


Figure 7: Reference Architectural Model Industrie 4.0 according to [1]

The *Virtual Representations* as discussed in the example is e.g. part of the 'Monitoring & Diagnostics' module of the operations domain ('Functional Viewpoint') of the IIC Reference Architecture [12] but the methodology is not restricted to it. ML trained models or the proposed program/query implementation of the configuration function are as well covered by the IIC operations domain. In addition, one can also locate the *Virtual Representation* in the underlying 'Implementation Viewpoint' as it acts as a kind of data source or a gateway, even though not between distinct networks but entities.

The Asset Administration Shell (AAS) concept of the 'Platform Industrie 4.0' [3] can act as a reference model for a *Virtual Representation* especially as it mentions the usage of RDF ontology concepts to clarify semantic meaning. An AAS acts as an interface between the virtual and the physical world, same as a *Virtual Representation*. Nevertheless, as discussed in [3], it is not yet discussed whether an AAS representing a physical objects requires a direct data connection. In the reference architecture in Fig. 7 the *Virtual Representation* is based in the integration and communication layer where the physical object is represented and accessible by higher level applications [1].

5 CONCLUSION AND FUTURE WORK

The concept of *Virtual Representations* serves as a mediator to bring unconnected physical objects quickly to the IoT. Needless to say, a simulated reflection cannot be as accurate as a directly observed measurement. Still, the advantage of the *Virtual Representation* is the low-cost deployment and its flexible ways to write the configuration into an encapsulated representation of a real-world object.

We did not focus on security related problems and yet disregarded *Virtual Representations* in other IoT protocols such as MQTT, CoAP, or OPC-UA. In addition, only a state-based request/response interaction is (yet) implemented, in contrast to data streams and events/notifications of many IoT applications. Future steps include the consisting modeling and description of the *Virtual Representation's* analytic logic, its interaction scheme and physical behavior in a machine-processable manner. The Hydra and openAPI vocabulary will be used to further document and explain the interface. Although we implemented an example execution environment as a HTTP Web server the common concept is not restricted to HTTP. We aim to provide similar projects for OPC-UA and publish/subscribe protocols like MQTT.

Virtual Representations can be one stepping stone for a faster introduction of the IoT to real-world production lines. We believe that the proposed method can reduce implementation costs and efforts and illustrated the approach with a very simple machine demonstrator. We further discussed how *Virtual Representations* fit in current reference works in order to clearly specify their characteristics.

Acknowledgement. The research and development project that forms the basis for this report is funded under project No. 01MD16015 (STEP) within the scope of the Smart Services World technology program.

REFERENCES

- [1] Plattform Industrie 4.0 and ZVEI. Reference Architectural Model Industrie 4.0 (RAMI 4.0), October 2016. <http://www.plattform-i40.de/140/Redaktion/EN/Downloads/Publikation/rami40-an-introduction.html>.
- [2] Sebastian R Bader. Automating the dynamic interactions of self-governed components in distributed architectures. In *European Semantic Web Conference*, pages 173–183. Springer, 2017.
- [3] Heinz Bedenbender, Alexander Bentkus, Ulrich Epple, Thomas Hadlich, Roland Heidel, Oliver Hillermeier, Michael Hoffmeister, Haimo Huhle, Markus Kiele-Dunsche, Heiko Koziolek, Steffen Lohmann, Marco Mendes, Jörg Neidig, Florian Palm, Stefan Pollmeier, Benedikt Rauscher, Frank Schewe, Martin Wollschlaeger, Wollschlaeger, Ingo Weber, and Bernd Waser. Industrie 4.0 Plug-and-Produce for Adaptable Factories: Example Use Case Definition, Models, and Implementation, 2017. <http://www.plattform-i40.de/140/Redaktion/EN/Downloads/Publikation/Industrie-40-%20Plug-and-Produce>.
- [4] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.
- [5] Fielding, Roy T and Taylor, Richard N. *Architectural styles and the design of network-based software architectures*. Irvine Doctoral dissertation. University of California, 2000.
- [6] Edward H Glaessgen and David Stargel. The digital twin paradigm for future nasa and us air force vehicles. In *53rd Struct. Dyn. Mater. Conf. Special Session: Digital Twin, Honolulu, HI, US*, pages 1–14, 2012.
- [7] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.
- [8] Andreas Harth, Craig A. Knoblock, Steffen Stadtmüller, Rudi Studer, and Pedro Szekely. On-the-fly integration of static and dynamic linked data. In *Proceedings of the Fourth International Conference on Consuming Linked Data*, volume 1034, pages 1–12, 2013.
- [9] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. Mobile edge computing—a key technology towards 5g. *ETSI White Paper*, 11(11): 1–16, 2015.
- [10] Felix Leif Keppmann and Maria Maleshkova. Smart Components for Enabling Intelligent Web of Things Applications. *INTELLI 2016*, page 128, 2016.
- [11] Jay Lee, Behrad Bagheri, and Hung-An Kao. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3: 18–23, January 2015.
- [12] Shi-Wan Lin, Bradford Miller, Jacques Durand, Graham Bleakley, Amine Chigani, Robert Martin, Brett Murphy, and Mark Crawford. The Industrial Internet of Things Volume G1: Reference Architecture, 2017. <http://www.iiconsortium.org/IIRA.htm>.
- [13] Maria Maleshkova, Carlos Pedrinaci, and John Domingue. Investigating web apis on the world wide web. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 107–114. IEEE, 2010.
- [14] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.
- [15] Goldman Sachs. The internet of things: Making sense of the next mega-trend. 2014. Available: www.goldmansachs.com/our-thinking/pages/internet-of-things/ [Online][Accessed on Jan 17, 2018], 2017.
- [16] Steve Speicher, John Arwe, and Ashok Malhotra. Linked data platform 1.0. <https://www.w3.org/TR/ldp/>, 2015. W3C Recommendation, accessed 30.06.2016.
- [17] Fei Tao, Jiangfeng Cheng, Qinglin Qi, Meng Zhang, He Zhang, and Fangyuan Sui. Digital twin-driven product design, manufacturing and service with big data. *The International Journal of Advanced Manufacturing Technology*, pages 1–14, 2017.